REST API

parse\_str(file\_get\_contents('php://input'), $requestData);

‘GET’

$books = BooksMapper::getBooks();

$serializedBooks = array();

//Go through all the books and add them to the serialized array

foreach ($books as $book) {

$serializedBooks[] = $book->jsonSerialize();

}

// var\_dump($serializedBooks);

header('Content-Type: application/json');

echo json\_encode($serializedBooks);

\*jsonSerialzie method

$obj = new StdClass;

$obj->ISBN = $this->getISBN();

Retrn &obj

‘POST’

case "POST":

$newBook = new Book();

$newBook->setPrice($requestData['price']);

$newBookId = BooksMapper::createBook($newBook);

header('Content-Type: application/json');

echo json\_encode($newBookId);

‘DELELE’

$isbnToDelete = $requestData['isbn'];

if(BooksMapper::deleteBook($isbnToDelete)){

header('Content-Type: application/json');

echo json\_encode($result);

‘PUT’

Mapper

private static $db;

static function initialize(string $className) {

self::$db = new PDOAgent($className);

}

$selectSql = "insert into Books values(:isbn, :author, :title, :price)";

self::$db->query($selectSql);

try{

self::$db->bind(":isbn", $newBook->getISBN());

return self::$db->lastInsertId();

delete returns boolean

$deleteQuery = "delete from books where isbn = :isbn";

try{

self::$db->query($deleteQuery);

self::$db->bind(":isbn",$isbn);

update

static function updateBook(Book $updateBook){

$updateQuery = "update books set title=:title, author=:author, price=:price where isbn=:isbn";

try{

self::$db->query($updateQuery);

self::$db->bind(":isbn", $updateBook->getISBN());

Page

'<a href= "' . $\_SERVER['PHP\_SELF'] . '?action=delete&isbn=' . $book->getISBN() . '">

Delete

</a>

Main Class,

RestClient::call('DELETE', array('isbn' => $\_GET['isbn']));

$jsonBook = RestClient::call('GET', array('isbn' => $\_GET['isbn']));

$jsonBook = json\_decode($jsonBook);

$bookUpdate = new Book();

$bookUpdate->setISBN($jsonBook->ISBN);

if($\_POST['edit'] == "true"){

$result = RestClient::call('PUT', $\_POST);

} else {

$result = RestClient::call('POST', $\_POST);

$jsonBooks = RestClient::call('GET', array());

$decodedBooks = json\_decode($jsonBooks);

$books = array();

foreach($decodedBooks as $book){

$newBook = new Book();

$newBook->setISBN($book->ISBN);

Make an object as standard class for endcoding json

$obj = new StdClass;

$obj->ISBN = $this->getISBN();

$obj->Title = $this->getTitle();

$obj->Author = $this->getAuthor();

$obj->Price = $this->getPrice();

//Return the standard class

return $obj;

<td><A HREF="'.$\_SERVER["PHP\_SELF"].'?action=delete&isbn='.$book->getISBN().'

">Delete</A></td>

Xml

$xml = new SimpleXMLElement('<xml/>');

$penXML = $xml->addChild('pen');

$penXML->addChild('color', $pen->color);

$xmlString = $xml->asXML();

$xmlPen = simplexml\_load\_string($xmlString);

Password

$hash = password\_hash($this->getPassword(), PASSWORD\_DEFAULT);

return password\_verify($passwordToVerify, $hash);

get information from session

$currentUser = $\_SESSION['loggedin'];

Page::welcome($currentUser);

Store data into session

session\_start();

//Set the logged in to true

$\_SESSION['loggedin'] = $user;

//Send the user to the user managment page

header('Location: Lab09-SWh-56789-welcome.php');

Header

header('Location: Lab09-SWh-56789-login.php');

Login manager

if(session\_id() == '' && !isset($\_SESSION)){

session\_start();

}

if(isset($\_SESSION["loggedin"])){

return true;

} else {

session\_destroy();

unset($\_SESSION);

return false;

}

Upload file

<form method = "POST" action="<?php echo $\_SERVER["PHP\_SELF"];?>" enctype="multipart/form-data">

<input type="file" name="fileToUpload" />

<input type="submit" value="Uplaod File" />

$contents = FileAgent::read($\_FILES["fileToUpload"]["tmp\_name"]);

$team = $teamParser->parseRoster($contents);

//Show pass the entire team to the page class for display

$teamPage->showRoster($team);

Read file

$fh = fopen($dataFile, 'r');

// Read the contents

$contents = fread($fh, filesize($dataFile));

// Close file handle

fclose($fh);

if(!$fh){

throw new Exception("File ". $dataFile . "is not found!");

}

FileAgent::write("data\oriolesSorted.csv", $team->\_\_toString());

Write

static function write($fileName, $contents) {

try {

$fh = fopen($fileName, 'w');

fwrite($fh, $contents);

fclose($fh);

if (!$fh)

throw new Exception("File $fileName could not be written!");

} catch (Exception $fe) {

Usort

$sortedTeam = usort($this->players, array($this, "compareTo"));

Write content with to string

public function \_\_toString(){

return "{$this->playerNum}, {$this->firstName}, {$this->lastName}, {$this->position}, {$this->bats}, {$this->throw}, {$this->age}, {$this->height}, {$this->weight}, {$this->birthPlace}";

}

Parser

self::$team = new Team();

$lines = explode("\n", $fileContents);

try{

for($i = 1; $i < count($lines); $i++){

$columns = explode(",", $lines[$i]);

if(count($columns) != 9)

throw new Exception("Invalid columns");

$fullName = explode(" ", $columns[1]);

$player = new Player(

$columns[0], $fullName[0], $fullName[1], $columns[2], $columns[3],

$columns[4],$columns[5],$columns[6],$columns[7],$columns[8]

);

self::$team->addPlayer($player);

}

} catch(Exception $e){

var\_dump($e->getMessage());

}

// var\_dump(self::$team);

usort(self::$team->players, array(self::$team,"compareLastName"));

return self::$team;